Classical Biomolecular Simulation on BG/L: Protein Science for Blue Gene

B.G. Fitch
R.S. Germain
Biomolecular Dynamics
and Scalable Modeling
http://www.research.ibm.com/bluegene/

August 14, 2002

Acknowledgements

- Blue Gene science/application team
  - Jed Pitera, Mike Pitman, Alex Rayshubskiy, Frank Suits, Bill Swope, Chris Ward, Yuri Zhestkov, Ruhong Zhou
- Collaborators
  - Bruce Berne (Columbia)
  - Vijay Pande (Stanford)
- Blue Gene/L hardware and system software teams

Outline

- BG Science overview (brief)
- Computational resource estimates for protein kinetics and thermodynamics
- Options for long range force evaluation
- Model calculations/estimates for the communications required for selected options (FFT, position globalization)
- Blue Matter overview
- Wrap-up

Science Mission Statement

We will use large scale biomolecular simulation to advance our understanding of biologically important processes, in particular our understanding of the mechanisms behind protein folding.

Advances in our understanding of biomolecular simulation required to achieve the scientific goals of the Blue Gene project can be applied to a variety of related problems including:

- Drug protein interactions (docking)
- enzyme catalysis (with hybrid quantum methods)
- structure refinement and scoring for database methods

Why Protein Folding?

- Proteins are linear polymers made from amino acids
- Amino acid sequence determines the final structure
- Structure is tightly related to protein function
- Nature has evolved proteins not only to provide specific functions, but to fold into the required conformation in a biologically relevant time scale
- Many proteins fold spontaneously*
- Diseases are associated with misfolding: e.g. Alzheimer's, BSE, Cystic Fibrosis
- Understanding protein folding mechanisms may help in developing self-assembling molecules for nanotechnology

*Some proteins require chaperones, and other forms of assistance; some are stabilized by internal chemical (disulfide) bonds and/or by association with other cellular structures or molecules.

Example of Protein (un)Folding

- β-hairpin in water at 900K for approximately 1ns
- requires $\approx$ 1 month of running time on a 375MHz Power3 CPU
- c-terminus of protein G

Protein Folding and Blue Gene

- **Folding Pathway Characterization** (sampling including Monte Carlo)
  - Describe the intermediate structures, the energy, entropy, free energy landscape analysis along the "reaction path", without interest in kinetics.
- **Folding Kinetics**
  - Describe the rates associated with the folding phases, the time spent in various states along the pathway.
- **Protein structure prediction** (comparative modeling, energy minimization)
  - By itself, structure prediction may not justify a large scale dynamical simulation effort. However, simulation might be used to refine structures produced by other methods.

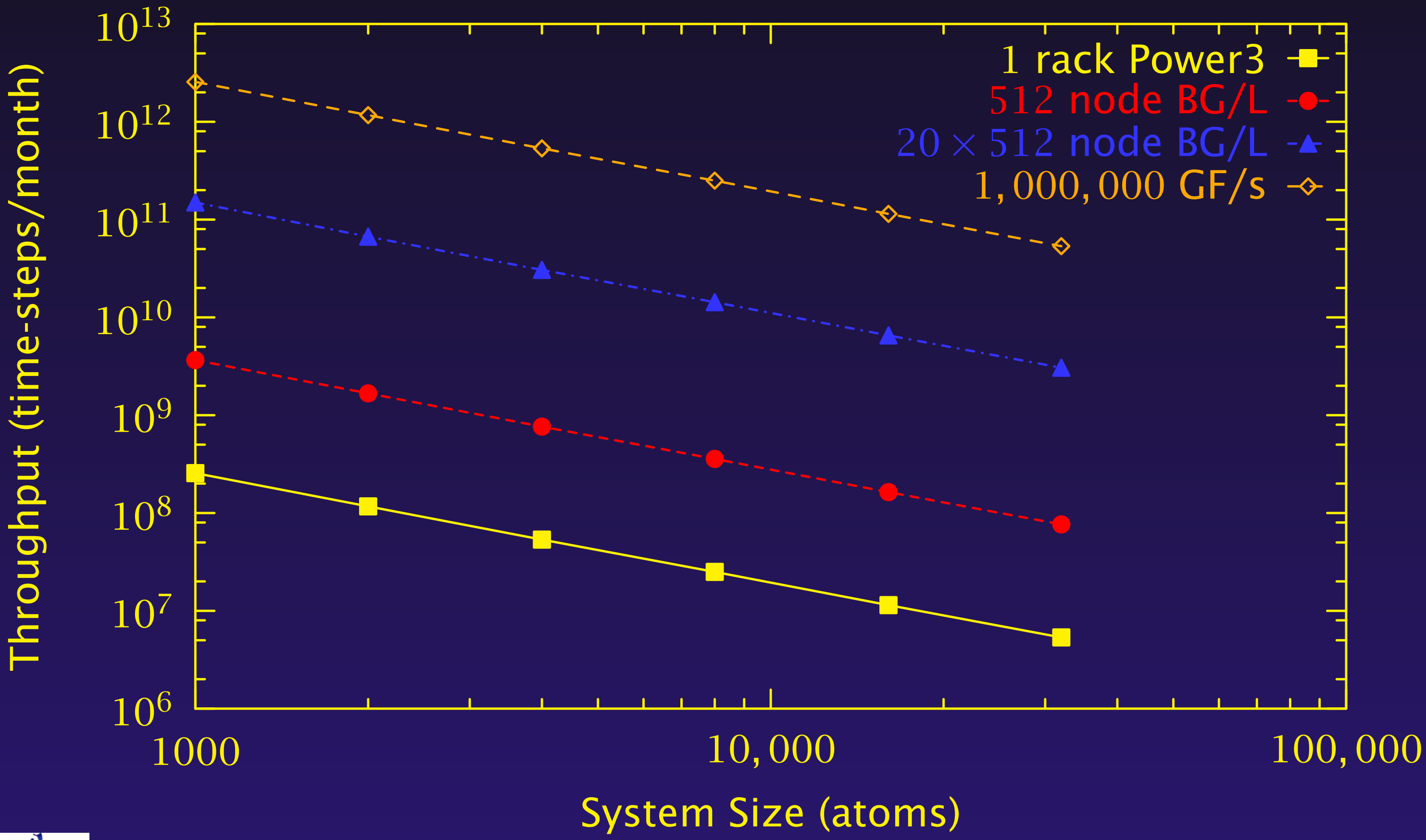
Science Goals

- Identifying disease-related processes whose critical stage pathway can be studied using large scale simulation
- Large scale studies of protein thermodynamics
- Long time scale studies of protein folding kinetics
- Connecting with experimental data
- Characterizing the models (force fields, water models) used in classical molecular simulation

Time Scales for Protein Folding Phenomena

Phenomenon	System/Size w/solvent	Time Scale	Time-step Count
peptide thermodynamics	α-helix, β-hairpin/4000	$0.1 - 1\mu$sec	10^8
peptide kinetics	β-hairpin/4000 atoms	5μsec	10^9
protein thermodynamics	$60 - 100$ residues/$20 - 30,000$ atoms	$1 - 10\mu$sec	10^9
protein kinetics	$60 - 100$ residues/$20 - 30,000$ atoms	500μsec	10^{11}

Extrapolated Computational Throughput



Assessing Bounds to Scalability

- Major components in molecular simulation
  - Bonded force evaluation
  - Real space non-bond force evaluation
  - Reciprocal space non-bond force evaluation
- Assess potential computational concurrency (neglecting communication overheads).
- Estimate communications overheads for key portions of algorithm.
- Use these estimates to prioritize application development effort.

Major Options for Non-bond Force Evaluation

- P3ME/PME (FFT-based)
- Ewald (and effective pair potential techniques)
- Multigrid
- Tree-codes (e.g. Periodic Fast Multipole Method)

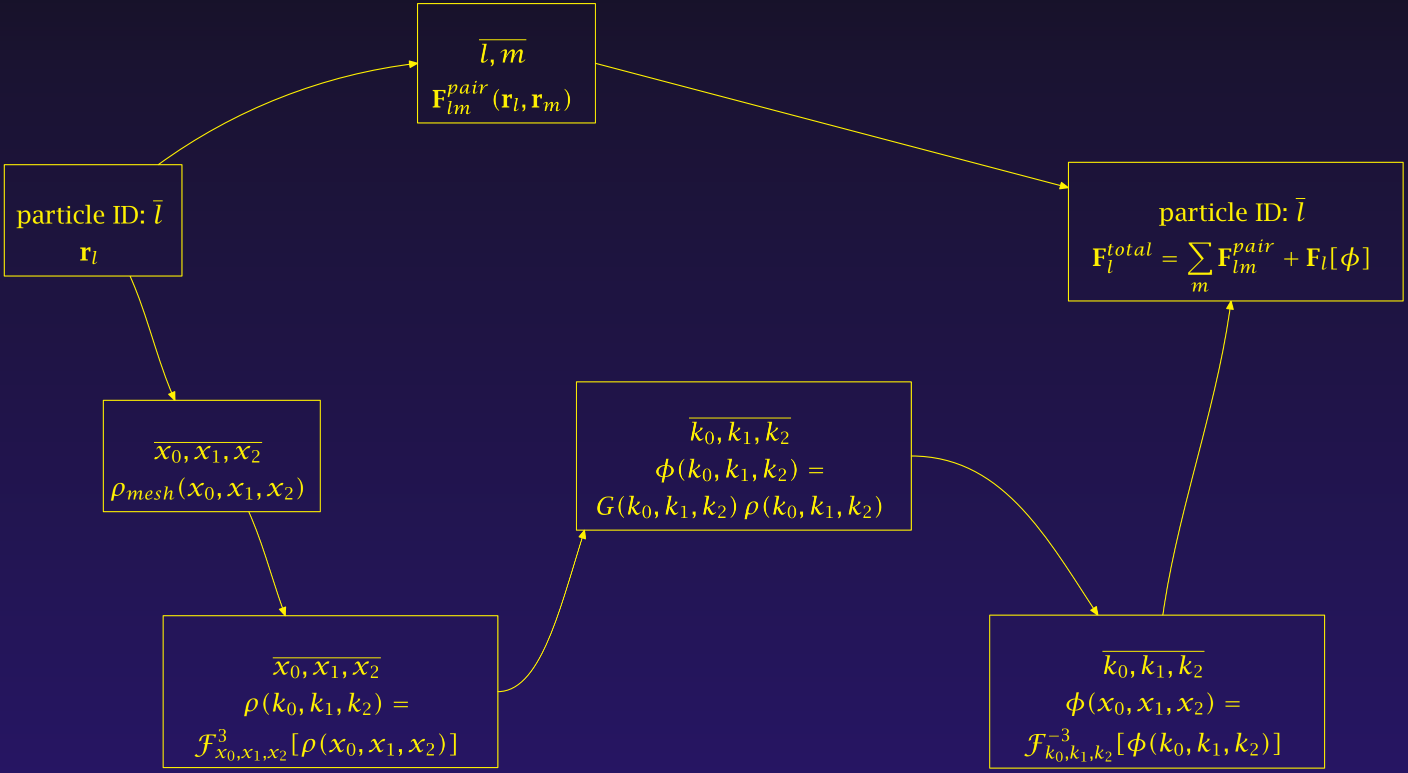
All of these typically require direct real space evaluation of pair interactions within some finite range.

Naive Scalability Model

- This model assumes globalization of positions, use of P3ME (neglecting required near neighbor force reductions)
- This is one of the decomposition strategies being evaluated.

$$\begin{aligned}
T_{ts} &= \frac{1}{p^3} \sum_i N_i^{udf} \, \tau_i^{udf} \\
&+ \frac{1}{p^3} N_{sites} \, \tau_{verlet} \\
&+ \left(\frac{N_{sites}}{p^3}\right) \frac{4}{3} \pi \, r_c^3 \, \rho \, \tau_{non-bond} \\
&+ \tau_{p3me}(p, N_{mesh}) \\
&+ \tau_{Globalize}(N_{sites}, p)
\end{aligned}$$

P3ME



particle ID: $\bar{l}$
$\mathbf{r}_l$
$\overline{l, m}$
$\mathbf{F}_{lm}^{pair}(\mathbf{r}_l, \mathbf{r}_m)$
particle ID: $\bar{l}$
$\mathbf{F}_l^{total} = \sum_m \mathbf{F}_{lm}^{pair} + \mathbf{F}_l[\phi]$
$\overline{x_0, x_1, x_2}$
$\rho_{mesh}(x_0, x_1, x_2)$
$\overline{x_0, x_1, x_2}$
$\rho(k_0, k_1, k_2) = \mathcal{F}^3_{x_0,x_1,x_2}[\rho(x_0, x_1, x_2)]$
$\overline{k_0, k_1, k_2}$
$\phi(k_0, k_1, k_2) = G(k_0, k_1, k_2)\, \rho(k_0, k_1, k_2)$
$\overline{k_0, k_1, k_2}$
$\phi(x_0, x_1, x_2) = \mathcal{F}^{-3}_{k_0,k_1,k_2}[\phi(k_0, k_1, k_2)]$

Estimate of Communications Requirements for Globalization of Positions

- Simple approach: Globalize all particle positions using the tree (raw bandwidth is $4\text{bits}/\text{cycle}$).
  - $T_{comm} = V_{positions}/R^{tree}_{bandwidth} \approx 2.8 \times 10^6$ cycles for 30,000 atom system.
  - Compare with lower bound for computational time using estimate of 340 cycles/non-bond interaction in a water box with a 10Å cutoff: 4.3×10^9 cycles. For a 512 node partition, assuming an even distribution of the load, $T_{comp} \approx 8.4 \times 10^6$ cycles.

Convolution (using 3D FFT)

$$\begin{tikzcd}
\boxed{\begin{array}{c}\overline{x_0,x_1,x_2}\\ \rho(x_0,x_1,x_2)\end{array}} \arrow[r] & \boxed{\begin{array}{c}\overline{x_0,x_1}\\ \mathcal{F}^1_{x_2}[\rho(x_0,x_1,x_2)]\end{array}} \arrow[r] & \boxed{\begin{array}{c}\overline{x_0,k_2}\\ \mathcal{F}^1_{x_1}[\rho(x_0,x_1,k_2)]\end{array}} \arrow[r] & \boxed{\begin{array}{c}\overline{k_1,k_2}\\ \mathcal{F}^1_{x_0}[\rho(x_0,k_1,k_2)]\end{array}} \arrow[dlll] \\
\boxed{\begin{array}{c}\overline{k_0,k_1,k_2}\\ \phi(k_0,k_1,k_2)=\\ G(k_0,k_1,k_2)\,\rho(k_0,k_1,k_2)\end{array}} \arrow[r] & \boxed{\begin{array}{c}\overline{k_1,k_2}\\ \mathcal{F}^{-1}_{k_0}[\phi(k_0,k_1,k_2)]\end{array}} \arrow[r] & \boxed{\begin{array}{c}\overline{x_0,k_2}\\ \mathcal{F}^{-1}_{k_1}[\phi(x_0,k_1,k_2)]\end{array}} \arrow[r] & \boxed{\begin{array}{c}\overline{x_0,x_1}\\ \mathcal{F}^{-1}_{k_2}[\phi(x_0,x_1,k_2)]\end{array}} \arrow[dl] \\
 & & \boxed{\begin{array}{c}\overline{x_1,x_2,x_3}\\ \phi(x_0,x_1,x_2)\end{array}} &
\end{tikzcd}$$

Model Estimate of Communication Time for 3D FFT on BG/L

- All-to-all estimation methodology thanks to P. Heidelberger, B. Steinmacher-burow, and A. Gara
- FFT of real-valued function on $N \times N \times N$ mesh using $p \times p \times p$ torus with each node handling $(N/p)^3$ mesh points.
- Each phase of the FFT requires all nodes to exchange data along a row or within a plane of nodes so that the N^2 1-D FFTs required can be computed locally.
- $T_{comm} = V_{received/node} / (n_{links} \cdot f_{utilized} \cdot R_{bandwidth} / \overline{n}_{hops})$
- For all-to-all within a row: $n_{links} = 2$ and $\overline{n}_{hops} = p/4$
- For all-to-all within a plane $n_{links} = 4$ and $\overline{n}_{hops} = p/2$
- For either row or plane:
  $T_{comm} = V_{received/node} \cdot p/(8 \cdot R_{bandwidth} \cdot f_{utilized})$ and
  $V_{received/node} = (N/p)^3 \cdot \mathrm{sizeof(double)}$
- $T_{comm} = N^3/(8 \cdot R_{bandwidth} \cdot f_{utilized} \cdot p^2)$

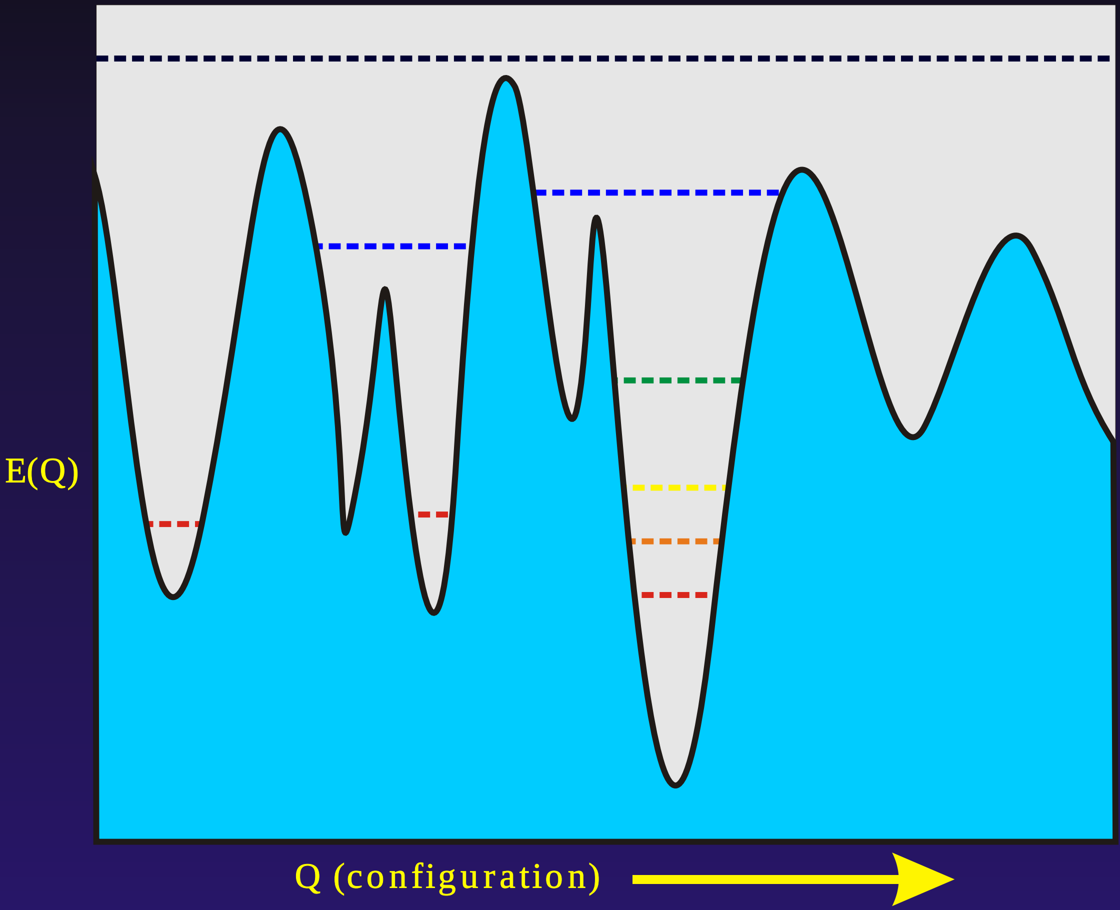
Estimates for 3D FFT on a $512 (8 \times 8 \times 8)$ node BG/L Partition

- Disclaimer: The following is not a prediction of real performance since such items as software overheads and memory hierarchy effects are not included in this crude estimate.
- Assuming $R_{bandwidth} = 2\mathrm{bits}/\mathrm{cycle}$, one phase (or one third) of a 3D FFT on a 128^3 mesh has a bandwidth limited time of $\approx 130,000$ cycles or $\approx 165,000$ cycles if 80% link utilization is assumed.
- Computation estimate: Using cycle count generated by vacpp compiler for BG/L (courtesy of T.J.C. Ward), 128 point real FFT on BG/L is estimated to take ≈ 5600 cycles so that the $128^2/p^3$ computed by each node should take $\approx 180,000$ cycles (for $p = 8$).
- Conclusion: It is worth investing effort in an implementation of the FFT-based P3ME method for periodic electrostatics.

Molecular Simulation Methodology that Extends Scalability

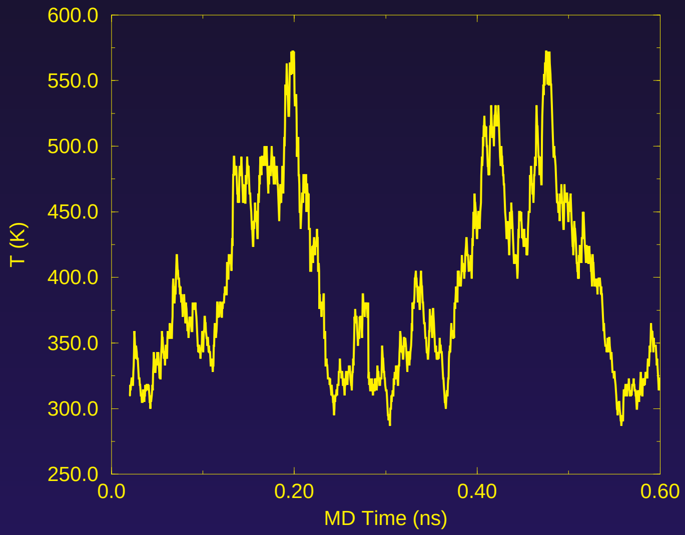
- To improve sampling efficiency for thermodynamic studies, a series of molecular dynamics (or MC) simulations are run on the same system, each run begin held at a different temperature.
- Periodically, an attempt is made to exchange configurations between different "replicas" using a Metropolis-style criterion.
- Depending on the system, anywhere from 32 to 128 or more replicas may be coupled together.
- With each replica running on a 512 node partition, good utilization can be obtained on 16K-64K nodes.

Replica Exchange Method

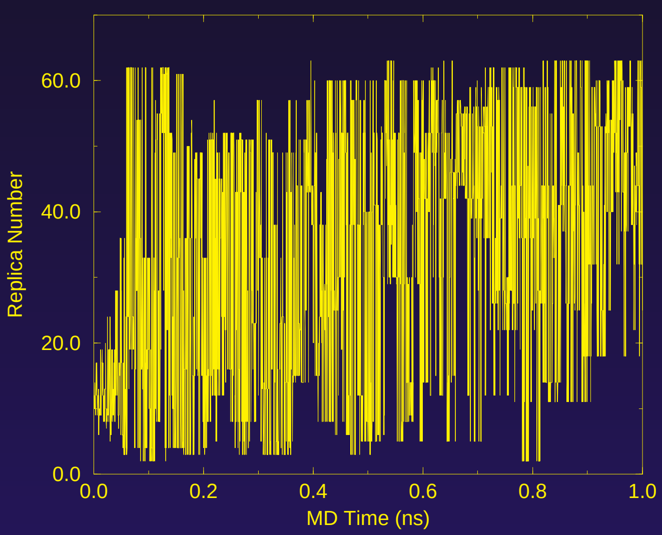


Replica Exchange Method Results

Temperature Trajectory of One Replica



Replica Trajectory in Temperature 310K



- 64 replicas used for β-hairpin in water ($\approx$5000 atoms)
- Temperature spans from 270 to 695 K

Drivers for the Application from the Science Program

- Study kinetics of solvated peptide/protein systems on very long scales (microseconds and longer).
- Characterize models(force fields, water models) used in classical biomolecular simulation
- Rigorous treatment of long range electrostatic interactions is a requirement.
- Need to gain confidence in validity of application code through use by Blue Gene Science team.
- All studies are statistical in nature (maximizing scientific throughput for a given experiment is the goal):
  - Multiple trajectories required for kinetics studies.
  - Multiple trajectories (coupled) required for efficient sampling in thermodynamic studies.

Implications for Application (from science)

- Relevant scalability is measured by speed-up on a fixed size problem rather than scalability with molecular system size.
- Treatment of long range electrostatic interactions implicates global exchange of information.
- Need to run efficiently on currently available platforms to support early science program.
- Efficient utilitization of large node count computational resources can be achieved by partitioning into independent or loosely coupled simulation runs.
- Simulating 1μsec trajectories requires $\approx 10^9$ time-steps—to complete a simulation of this magnitude within a month of running time requires individual time-steps to be computed within ≈ 3msec.
- Multiple force field support

Drivers for the Application from the Blue Gene Platforms

- 512–64K node counts (for BG/L)
- modest memory per node
- need to be conscious of interconnect topology
- chip architecture (two CPUs, pipelined, double FPU, etc.)

Implications for Application (from platforms)

- Fine-grained concurrency in application decomposition
- Short code paths (particularly from network to application logic)
- Minimize memory footprint of application (stay resident as high in memory hierarchy as possible)
- Application requires compiler and/or hand optimization to fully utilize chip (including inlining, unrolling, instruction scheduling, ...)
- Efficient application checkpoint/restart capability

Software Engineering Considerations

- Layering to facilitate separation of MD complexities from those of parallel software
- Black box testing comprising both regression and validation of molecular dynamics functionality
- Need to target machines of dramatically different sizes and configurations
- Need to support a variety of MD techniques and experimental environments
- Enabling customized builds of application kernels containing only desired components
- Leveraging existing MD packages where possible for problem setup

High Level Design Decisions

- Strip as much domain (chemical) knowledge as possible out of the parallel kernel
- Datagram-based i/o — no dependency on filesystem availability
- Active message (packet) layer as basis for communications
- Customized generation of runtime kernel in *standard* C++
- Reliance on efficient inlining to allow logical application layering
- Support for application-based subpartitioning where it makes sense, e.g. loosely coupled trajectories for replica exchange

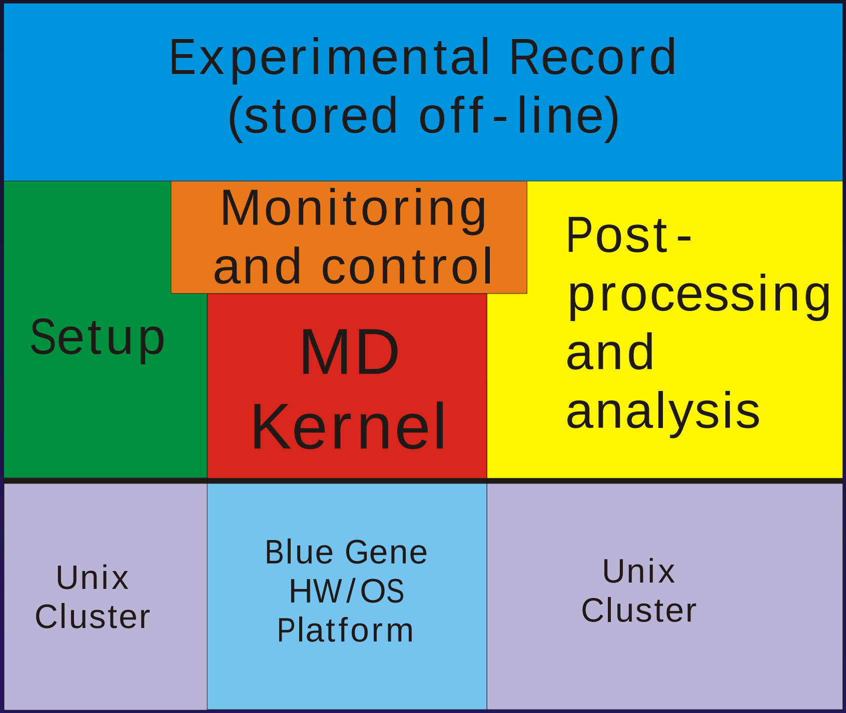
Blue Matter

- Application platform for the Blue Gene Science program
- Prototyping platform for exploration of application frameworks suitable for cellular architecture machines
- Blue Matter comprises all of the necessary application components–those that run on the computational core and those that run on the host
- Not an evolution from existing code

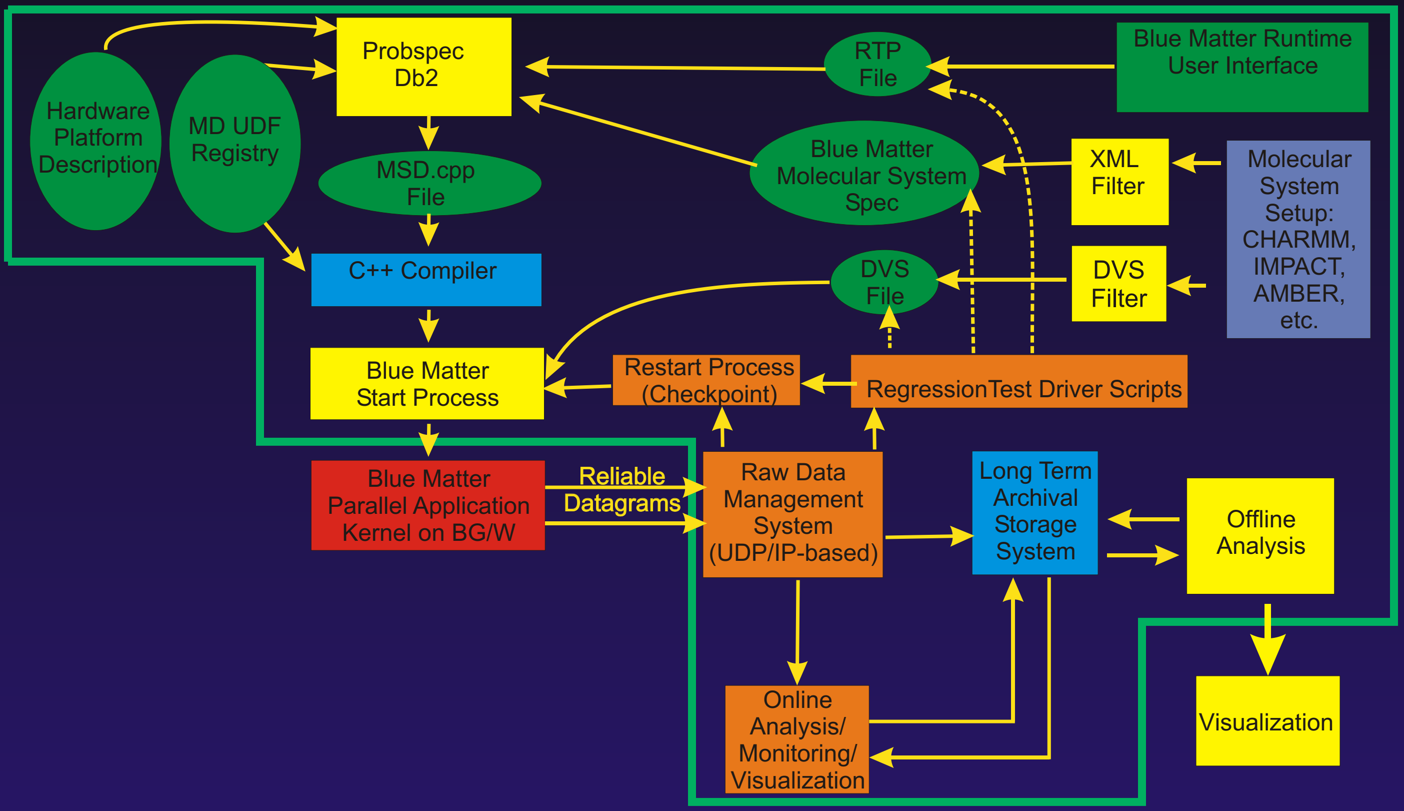
Blue Matter

- Collection of modules including the following:
  - Generator for MD core engine (massively parallel, minimal in size, runs on BG/L)
  - Utility programs to import force field assignments from other packages, manage molecular system specifications in database, etc.
  - Monitoring and analysis tools to analyze MD trajectories, etc.
- Framework for “black box” testing of molecular simulation function
  - Enable algorithmic and tuning explorations while preserving application semantics
  - Attempt to minimize impact of coding explorations on science team

Blue Matter Overview



Blue Matter Dataflow


Hardware Platform Description
MD UDF Registry
Probspec Db2
MSD.cpp File
C++ Compiler
Blue Matter Start Process
Blue Matter Parallel Application Kernel on BG/W
Reliable Datagrams
RTP File
Blue Matter Runtime User Interface
Blue Matter Molecular System Spec
XML Filter
Molecular System Setup: CHARMM, IMPACT, AMBER, etc.
DVS File
DVS Filter
Restart Process (Checkpoint)
RegressionTest Driver Scripts
Raw Data Management System (UDP/IP-based)
Long Term Archival Storage System
Offline Analysis
Online Analysis/ Monitoring/ Visualization
Visualization

Application Programming Techniques

- Separate computation from communication
- Currently:
  - Identify data-dependencies “by hand”
  - At each processing stage, identify appropriate partitioning keys
  - Program flow is a directed acyclic graph (DAG) where the work for each node may be partitioned
  - User implements “call-backs” as methods of classes that are passed as template parameters to the parallel application framework that handles data transport
- Areas for future research:
  - High level specifications of applications/algorithms
  - Partial or complete automation of data dependency analysis
  - Partial or complete automation of customized parallel framework

Application Programming Example: Plan for P3ME/Convolution

- C++ template class parameterized by
  - processor mesh dimensions
  - dimensions of charge mesh
  - Serial 1D FFT function object
  - Kernel function object (Green's function)
- Compiles to BG/L packet layer
- Active packet message handler is a memory put and also handles synchro-nization

Options for Parallel Decomposition Targeting BG/L

- Global force reduction — replication of dynamics propagation
- Globalize positions — double computation of forces, P3ME issues
- Globalize positions with nearest neighbor force reduction with approximate volume decomposition — supports P3ME
- Globalize positions with near neighbor (cutoff radius) force reduction with approximate volume decomposition — supports P3ME, avoids double computation of forces

Load Balancing Issues

- p nodes implies imbalance smaller than $1/p$ to maintain scalability
- $O(1\text{msec})$ time-step execution time means "real-time"-like programming techniques required
- currently based on global information
- future work involves approach using near neighbor communication

Compiler-related Work

- Work with Toronto compiler group to improve code generation (floating point unit utilization) by providing computational kernels from molecular simulation application.
- Toronto group (Mark Mendell) modified back-end to product compiler to:
  - spot opportunities for inlining better
  - increase window size for instruction reordering optimization
- Compiler modifications target BG/L, but can be used to generate code for Power3/4 targets
- Source code changes:
  - structure a loop for 3-way unroll
  - increase the independence of parts of the loop
- Measured improvements of 30-40% in execution times of selected kernels on Power3 platform obtained by source code changes (T.J.C. Ward) and use of modified back-end.

Wrap-up

- BG/L communications capabilities enable use of straightforward approaches for long range force evaluation for an interesting range of node counts.
- Areas for further work
  - Communications sizings for additional alternatives (to explore their scalability limits).
    - Ewald
    - treecodes (e.g. periodic fast multipole)
    - multigrid
  - More detailed sizings of communication and computation for selected molecular simulation kernels using cycle accurate simulators and other tools as they become available.